INFORMATION AND CONTROL 17, 203–215 (1970)

Note on Tape Reversal Complexity of Languages*

T. KAMEDA** AND R. VOLLMAR

*Institut für Mathematische Maschinen und Datenverarbeitung****, *Universität Erlangen-Nürnberg, Germany*

The number of tape reversals required for the recognition of a set of inputs by a 1-tape Turing machine (TM) has been proposed before as a measure of complexity of the set. In this paper, a tape reversal complexity is defined in terms of a multi-tape off-line TM, and the effect on the reversal complexity of reducing the number of available tapes is investigated.

It is shown that if a set is recognizable by a multi-tape TM with no more than $R(n)$ tape reversals, where n is the input length, then the set is recognizable by a 2-tape off-line TM with no more than $6R(n)$ tape reversals.

It is also shown that if a set is recognizable by a multi-tape TM within time $T(n)$, then the set is recognizable by a 1-tape off-line TM with no more than $T(n)$ tape reversals. For the special case $T(n) = n$, a language is exhibited which requires this bound.

Upper bounds on the number of reversals necessary for the recognition of a number of well-known classes of languages are also obtained.

I. INTRODUCTION

The first attempt to classify computations according to the number of necessary tape (or head) reversals was made by Trachtenbrot (Bečvář, 1965). The so-called *tape reversal complexity* has recently been further investigated (Fischer, Hartmanis, and Blum, 1968; Fischer, 1968; Hartmanis, 1968).

Fischer and Hartmanis discuss the reversal complexity of 1-tape Turing machine (TM) computations only. In this paper we first define a reversal complexity of a set of inputs in terms of a multi-tape off-line TM. We then relate the reversal complexity on a 2-tape TM and that on a 1-tape TM with

* This work was supported in part by Deutsche Forschungsgemeinschaft (ER 16/14, Ha 417/7) and in part by the National Research Council of Canada under grant No. A-4315.

** Present address: Department of Electrical Engineering, University of Waterloo, Waterloo, Ontario, Canada.

*** 8520 Erlangen, Egerlandstrasse 5. Director: Prof. Dr. W. Händler.

other complexities (Theorems 1-3). In this connection, the reversal complexity on a TM with stationary moves has, in general, different properties from that on a TM without stationary moves.

Finally the upper bounds on the reversal complexities of a number of well-known languages are investigated. To spare the reader the trouble of going through elaborate and water-tight definitions and proofs, we shall treat the topic rather informally, trusting the reader to fill in the details if necessary.

II. Preliminaries

A *k-tape* TM M over a finite input alphabet Σ consists of a *finite control* which can assume a state out of a finite *state set* $S = \{q_0, q_1, ..., q_r\}$, and k *tapes* which are chains of squares infinite in both directions. The finite control has an access to each tape through a *read-write head* (or simply a *head*) which is placed on one of the squares at any given time. M is started in the designated *start state* q_0 with an input from Σ^* (the set of all finite length words over Σ) written on one of the tapes. The head on this tape is initially scanning the leftmost symbol of the input and each of the other tapes initially consists only of *blanks* B ($B \notin \Sigma$), with the head scanning one of them. The *transition function* δ, which is incorporated in the finite control, determines (depending on the present state $q \in S$ and the symbol from the *tape alphabet* $\Gamma(\Gamma \supset \Sigma \cup \{B\})$ under each head) the *move*, that is to say, the next state, the new symbol from $\Gamma - \{B\}$ to replace the symbol under each head, and the movement of each head. The movement of each head is specified by $d(d = -1, 0, 1)$, which says that the head in question be moved d squares to the right. The states in $F \subset S$ are designated as *final states*. If M reaches a configuration for which δ is not defined, M is said to have *halted*. An input is *accepted* by M, iff M halts in a final state. The set of all inputs accepted by M is called the *set recognized by* M. The TM M is designated by $M = (S, \Sigma, \Gamma, \delta, q_0, F)$ and sometimes called a k-tape *off-line* TM, in contrast to an *on-line TM*, in which input is written on a separate *input tape* that has a one-way read-only head. In this paper, by a k-tape TM we shall mean a k-tape off-line TM as defined above.

Let $\mathcal{T}(k)$ be the class of all k-tape TM's. For our later discussion, we need to define a subset $\mathcal{T}'(k)$ of $\mathcal{T}(k)$: $M \in \mathcal{T}(k)$ belongs to $\mathcal{T}'(k)$, iff M does not have *stationary moves*, i.e., d never takes the value 0.

A TM is said to be $T(n)$ *time bounded*, if it halts, for all inputs of length n, within $T(n)$ moves. A TM is said to be $R(n)$ *reversal bounded*, if it halts, for all inputs of lenth n, before the sum of head reversals on all tapes exceeds

$R(n)$. A TM M *defines time* $T(n)$, if M is $T(n)$ time bounded and for each n there is an input of length n for which M makes exactly $T(n)$ moves before it halts. Similarly, a TM M *defines reversal* $R(n)$, if M is $R(n)$ reversal bounded and for each n there is an input of length n for which M makes exactly $R(n)$ reversals before it halts.

In this paper we consider only TM's which halt for all inputs. Stated in terms of the sets recognized by these TM's, we consider only the recursive sets.

A set $L \subset \Sigma^*$ is said to be $T(n)$ *time recognizable* ($R(n)$ *reversal recognizable*), if there is a $T(n)$ time bounded TM ($R(n)$ reversal bounded TM) which recognizes L. We sometimes say that a set is, for example, $T(n)$ *time recognizable on a* TM *in* $\mathscr{T}(k)$ *or* $\mathscr{T}'(k)$, the meaning being self-evident.

One might be tempted to define the *time complexity*, for example, of a set $L \subset \Sigma^*$ as a "minimum" $T(n)$ such that L is not $T_1(n)$ time recognizable for any $T_1(n)$ satisfying $\inf_{n\to\infty}(T_1(n)/T(n)) = 0$. But this attempt fails, because there is an L for which no such $T(n)$ exists (Blum, 1967). Therefore by a *time* or *reversal complexity*, we shall vaguely mean the property of a set which is reflected in the recognition time or the number of reversals necessary for its recognition.

III. Main Results

Our first result relates the reversal complexity on a multi-tape TM with that on a 2-tape TM.

Theorem 1. *A set of inputs recognized by an* $R(n)$ *reversal bounded* TM *in* $\mathscr{T}'(k)$, *where* $k \geqslant 1$, *is recognized by a* $6R(n)$ *reversal bounded* TM *in* $\mathscr{T}'(2)$.

Proof. The cases $k = 1, 2$ are trivial. Therefore we assume $k \geqslant 3$. Let M be a given $R(n)$ reversal bounded TM in $\mathscr{T}'(k)$. We describe a $6R(n)$ reversal bounded TM M' in $\mathscr{T}'(2)$, which simulates the moves of M. Let T_1 and T_2 be the two tapes of M'. Each of them has k tracks corresponding to the k tapes of M. T_1 simulates the tapes of M, on which the heads are moving to the right, and T_2 simulates the tapes of M, on which the heads are moving to the left.

Suppose, without loss of generality, that the head on the ith ($1 \leqslant i \leqslant k$) tape of M reverses the direction of its move from right to left. Before the reversal, the ith tape was simulated by T_1, and after the reversal, by T_2. But to continue the simulation, the ith track of T_2 must be updated by copying

the contents of the ith track of T_1. To do this, the squares on T_1 and T_2 under scan are marked and then both heads are moved all the way to the right, until both of them reach the blank portion of the tapes. Then the two heads are moved to the left, whereby the contents of the ith track of T_1 is copied to the ith track of T_2. When both heads reach the blank portion at the left ends of the tapes, the heads are brought back to the marked squares. During the above process, the head of T_1 undergoes 2 reversals and that of T_2, 4 reversals. Thus altogether 6 reversals suffice to simulate a reversal of M. ∎

At present we don't know if the coefficient 6 can be made smaller. It is interesting to compare the above result with the corresponding result for time complexity. Hennie and Stearns (1966) have shown that if a set is $T(n)$ time recognizable, then it is recognized by a $T(n)$ log $T(n)$ time bounded TM in $\mathscr{T}(2)$.

Next theorem gives a relation between the reversal complexity on a 1-tape TM and time complexity. First we cite a result by Fischer (1968).

LEMMA 1 (Fischer). *For any constant $\alpha > 0$, if a set is recognized by an $R(n)$ reversal bounded* TM *in $\mathscr{T}(1)$, it is recognized by an $\alpha R(n)$ reversal bounded* TM *in $\mathscr{T}'(1)$.*

THEOREM 2. *Any $T(n)$ time bounded* TM *in $\mathscr{T}(k)$, $k \geqslant 1$, can be simulated by a $T(n)$ reversal bounded* TM *in $\mathscr{T}'(1)$.*

Proof. Let M be a given $T(n)$ time bounded TM. We construct a $T(n)$ reversal bounded TM M' with one tape. The tape of M' has one track for each tape of M. Each track is divided into two levels, the lower level for tape symbols of M, the upper level for a marker. The marker on each track remembers the head position on the corresponding tape of M. It is easy to see that each move of M can be simulated by at most 2 reversals of M'. Thus if M is $T(n)$ time bounded, M' is $2T(n)$ reversal bounded. By Lemma 1, we can construct a $T(n)$ reversal bounded TM in $\mathscr{T}'(1)$, which simulates M'. ∎

We need the next lemma by Hartmanis (1968) to prove the following theorem.

LEMMA 2 (Hartmanis). *Let M be a* TM *in $\mathscr{T}(1)$, and suppose M defines time $T(n)$ and reversal $R(n)$. If $T(n) \geqslant c'n^2$ for some constant c', then $T(n)$ and $R(n)$ satisfy the relation $T(n) \geqslant R(n) \geqslant cT(n)^{1/2}$, where c is a constant.*

Proof. Hartmanis proves the result for on-line TM's, but the same proof applies here also. ∎

THEOREM 3. *There exists a set, for which $T(n)$ reversals in Theorem 2 are necessary.*

Proof. There are $T(n)$ time recognizable sets which require at least $c'T(n)^2$ time for recognition on a 1-tape TM for some constant c'. The set

$$L_1 = \{w\phi w^T / w \in \{0, 1\}^*, \phi \notin \{0, 1\}\},$$

where w^T is the mirror image of w, is an example (Hartmanis, 1968a). Let L be such a set and suppose M in $\mathscr{T}'(1)$ recognizes L. Let M define time $T_1(n)$ and reversal $R(n)$. Then because of the assumption about L,

$$T_1(n) \geqslant c'T(n)^2$$

for some constant c'. Applying Lemma 2, we get

$$R(n) \geqslant cT_1(n)^{1/2} \geqslant c(c'T(n)^2)^{1/2} = c''T(n),$$

where c'' is a constant. No matter how we choose M, the reversal $R(n)$ defined by M must satisfy the above relation. The constant c'' can be made unity by Lemma 1. ∎

Using the concept of the crossing sequence (Hennie, 1965), it can be directly shown that the set L_1 defined above requires at least cn reversals for some constant c for its recognition on a 1-tape TM.

IV. BOUNDS ON THE REVERSAL COMPLEXITY OF SOME LANGUAGES

In this section we try to find good upper bounds on the number of reversals necessary to recognize any language in some well-known classes. As is expected, an algorithm which has been developed to minimize the recognition time usually requires a relatively small number of reversals.

Suppose a language has the property that if it is $R(n)$ reversal recognizable on a TM in a certain class, it is also $\alpha R(n)$ reversal recognizable on some TM in the same class for *any* constant $\alpha > 0$. (cf. Lemma 1). Then we say that the language is $\sim R(n)$ *reversal recognizable*, and also that "*speed up*" *applies* to the recognition of that language. If none of the known "speed up" theorems apply, we have to say that *the language is $cR(n)$ reversal recognizable for some constant c.*

(1) Regular languages (rl): No reversal is needed (Rabin and Scott, 1959).

(2) Linear context-free languages (lcfl) (Ginsburg, 1966): Kasami's (1967)

algorithm requires only $c'n$ reversals on a 1-tape on-line TM with stationary moves for some constant c'. This TM can be simulated by a 1-tape off-line TM with cn reversals for some constant c (*F*ischer, 1968). Applying "speed up" (Lemma 1), we obtain the bound $\sim n$.

(3) Deterministic context-free languages (dcfl) (Ginsburg, 1966): Since every dcfl is $2n$ time recognizable (Aho, Hopcroft, Ullman, 1968), it is $\sim n$ reversal recognizable on a 1-tape TM by our Theorem 2 and Lemma 1.

(4) Context-free languages (cfl): The algorithms developed by Younger (1967) and Torii, Kasami and Ozaki (1966) can be implemented on a TM without stationary moves, so that only cn reversals are required for some constant c. (See Appendix I). Therefore cn is also a bound for a TM with stationary moves.

(5) 2-way deterministic pushdown automaton languages (2DPDAl) (Gray, Harrison, Ibarra, 1967): The algorithm by Aho et al. requires cn reversals in their implementation on a TM. (See Aho et al., 1968, Section V, not VI.) A cn^2 reversal bounded TM without stationary moves can be found to do the required job. (See Appendix II.)

(6) 2-way pushdown automaton languages (2PDAl) (Gray et al., 1967): The algorithm by Aho et al. (1968) requires cn^2 reversals in their implementation on a TM. A cn^3 reversal bounded TM without stationary moves can be found to do same job. (See Appendix II.)

The above results are summarized in Table I. For comparison, we also list

TABLE I

Bounds on the number of reversals

	Stationary moves allowed	Stationary moves not allowed	
rl	0^*	0^*	0^*
lcfl	$\sim n$	$\sim n$	$\sim n^*$
dcfl	$\sim n$	$\sim n$	$\sim n^*$
cfl	cn	cn	$\sim n^3$
2DPDAl	cn	cn^2	$\sim n^2 \log n$
2PDAl	cn^2	cn^3	$\sim n^4$

* is a tight bound, c is some constant.

the bounds on reversals necessary on a 1-tape TM. They have been obtained from the corresponding algorithms mentioned above and Theorem 2. Because of Lemma 1, we don't have to make distinction between recognitions on 1-tape TM's with and without stationary moves.

The set L_1 we defined before is a dcfl and at the same time a lcfl. It requires at least $\sim n$ reversals on a 1-tape TM.

The 1-tape TM constructed in (Taniguchi and Kasami, 1969) for the purpose of minimizing the recognition time of a cfl requires also $\sim n^3$ reversals (cf. Table I). This and other observations make us suspect that the "minimum" time and "minimum" reversals may be attained on the same TM. Whether this is in general true or not is a topic of our further research.

APPENDIX I: PROOF THAT EVERY CFL IS *cn* REVERSAL RECOGNIZABLE ON A TM WITHOUT STATIONARY MOVES

Let $G = (V, \Sigma, P, \sigma)$ be a context-free grammar, where V is the set of *terminal* and *nonterminal symbols*, Σ is the set of terminal symbols, P is the set of *productions*, and σ is the *sentence symbol* (see Ginsburg, 1966). Without loss of generality, we assume each production is of the form $X \to YZ$ or $X \to a$, where $X, Y, Z \in V - \Sigma$ and $a \in \Sigma$.

Let $w = a_1 a_2 \cdots a_n$ ($a_\ell \in \Sigma$, $\ell = 1, 2, \ldots, n$) be an input word to our TM recognizer. We assume endmarkers are attached to a_1 and a_n. For i and j such that $1 \leqslant i \leqslant j \leqslant n$, we define

$$N(i,j) = \{X \in V - \Sigma \mid X \overset{*}{\Rightarrow} a_i a_{i+1} \cdots a_j\} \tag{1}$$

Then we have $w \in L(G)$ iff $\sigma \in N(1, n)$, where $L(G)$ is the language generated by G. Younger's algorithm makes use of the relation

$$N(i,j) = \bigcup_{i \leqslant t < j} \{X \in V - \Sigma \mid X \to X_1 X_2 \in P \wedge X_1 \in N(i,t) \wedge X_2 \in N(t+1, j)\} \tag{2}$$

where $i < j$.

We now construct an $n \times n$ *recognition matrix*, whose (i, j) element where $i \leqslant j$, is going to be $N(i, j)$. (See Fig. 1(a)). All the entries below the main diagonal will be left blank. We fill the rest of the matrix in the order indicated by the arrows in Fig. 1(a): First, the entries on the main diagonal are filled using $N(i, i) = \{X \in V - \Sigma \mid X \to a_i \in P\}$. The other entries are filled using the relation (2). To be more specific, $N(i, j)$ can be computed by scanning the matrix in two directions as indicated in Fig. 1(b). The little circles

at the tails of the arrows indicate the starting positions. By the time the scan along the horizontal arrow reaches the (i, j) entry, we will have computed $N(i, j)$, so that it can be entered there. But we keep scanning blank entries until we reach the arrow heads for the reason which will become clear when we describe an implementation on a TM. Note that the horizontal and the vertical scans involve the same number of entries, and that when the scans are finished we are ready to compute the next entry, namely $N(i + 1, j + 1)$.

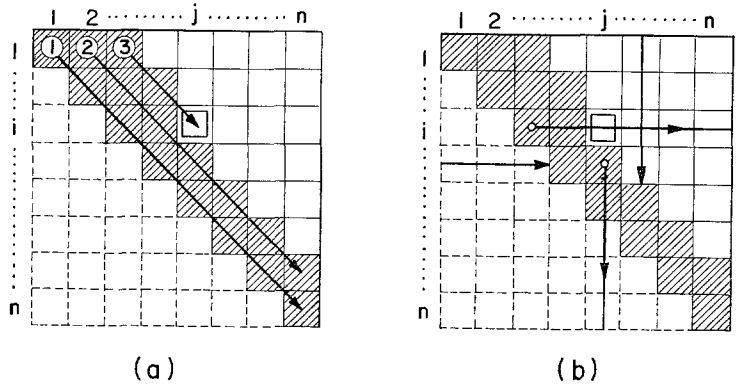

FIG. 1. Recognition matrix for a cfl.

In order to overcome some technical difficulty, when n is odd, we add a dummy row at the bottom and a dummy column at the right end of the recognition matrix, so that the size n' of the matrix is always even, i.e., $n' = 2[(n + 1)/2]$, where [ ] indicates the integer part of the number inside.

We now try to construct a TM, $M \in \mathscr{T}'(3)$, which performs the computation described above. We copy the matrix on two tapes T_1 and T_2. First slice the matrix horizontally and copy the 1st row (all the n' entries) on T_1, followed by the 2nd row, and so forth. Then slice the matrix vertically and copy the 1st column (all the n' entries), top to bottom, followed by the 2nd column, and so forth. Each row or column copied on T_1 or T_2 will be refered to as a *block*. For $\ell = 1, 2$, we use $T_\ell(i, j)$ to denote the square corresponding to the (i, j) entry of the matrix.

Initially the input word is written on the third tape T_3, called the *input tape*. Using T_3, M marks off $n'/2$ squares at the beginning of T_3. This can be done by scanning the n' squares (starting at the square with a_1 on it) on T_3, checking off the leftmost and the rightmost unchecked squares on that segment during each scan. Let the heads on T_1, T_2, and T_3 be named

H_1, H_2, and H_3, respectively. During these scans, H_3 reverses its direction $n'/2$ times. All this time, H_1 and H_2 move in exact synchronism with H_3, reversing their directions only when H_3 does. Thus it is easy to mark off $n'/2$ squares on T_1 as well. When this is done all the heads are moved to the left until H_3 gets on the first input symbol a_1. At this time H_1 and H_2 mark the squares they are on, and from now on these squares will be called the *left end* of T_1 and T_2, respectively.

Next we initialize the computation of the entries of the recognition matrix. The squares on T_1 and T_2 corresponding to the main diagonal of the matrix are filled by reading symbols from T_3 and using the relation

$$N(i, i) = \{X \in V - \Sigma \mid X \to a_i \in P\}.$$

First of all $T_1(1.\ 1)$ and $T_2(1, 1)$ are filled immediately, for H_3 is scanning a_1 and H_1 and H_2 are on $T_1(1, 1)$ and $T_2(1, 1)$, respectively. H_1 and H_2 have to move $n' + 1$ squares to the right before they reach $T_1(2, 2)$ and $T_2(2, 2)$, respectively, corresponding to the next main diagonal element, namely the (2, 2) entry. While this is taking place, H_3 moves only $n'/2 + 1$ squares to the right and turns back toward the left end of T_3, where it reverses its direction again, so that it will be scanning a_2 at exactly the same time as H_1 and H_2 are on $T_1(2, 2)$ and $T_2(2, 2)$, respectively. After the first half of the diagonal elements are computed in this manner, H_3 stays only on the right half of the input word. When all the main diagonal elements are computed, H_3 moves to the right without reversing its direction at all. (M has no use for H_3 anymore). So far the total number of head reversals has been proportional to n'.

Now M moves H_1 and H_2 to the left end of T_1 and T_2, respectively.

From now on computations can be carried out in a more systematic manner. They proceed as indicated in Fig. 1. The following steps are taken for $i = 1, 2, \ldots, n$.

(A) At the beginning of the computation of the entries on a diagonal, H_1 is on $T_1(1, 1)$ and H_2 is on $T_2(1, i)$.

(B) All the entries on any diagonal are computed without reversal of any head. Newly computed entries are written on T_1 only.

(C) When (B) is over, H_1 and H_2 reverse their directions and go back to the positions stated in (A). During this process the newly computed entries are copied to the corresponding squares on T_2.

(D) If $i = n$, the computation terminates. Otherwise H_2 is moved right to $T_2(1, i + 1)$, while H_1 is first moved right by $n'/2 + 1$ squares and then back to $T_1(1, 1)$.

The above four steps are repeated $n - 1$ times, at the end of which we will have computed $N(1, n)$ and the input is accepted iff $\sigma \in N(1, n)$. Note that in (B) above, even though H_2 "lags" behind H_1 by one square, the computations can be carried out as described before, by remembering the contents of the last square scanned by H_1.

None of the steps above requires more than a fixed number of reversals. Therefore it follows that the total number of reversals is bounded by cn, for some constant c.

APPENDIX II: PROOF THAT EVERY 2PDA1 (2DPDA1) IS cn^3 (cn^2) REVERSAL RECOGNIZABLE ON A TM WITHOUT STATIONARY MOVES

In order to avoid a long description of the algorithm by Aho et al. (1968), we assume that the reader is familiar with their paper and shall use their notations without even defining them. (The pages given in this appendix refer to those of their paper). Here we attempt a brief sketch of the proof.

We shall start with the *recognition matrix* (p. 192) with the slight modification that each main diagonal element will be marked prior to the filling of the matrix.

Just like in their implementation, our TM recognizer M has one input tape and working tapes. (We need only two working tapes because M can write on the input tape.) The roles of these tapes are just about the same as in their paper, except that the way the recognition matrix is stored in the two working tapes, T_1 and T_2, is different. The order *we* adopt is indicated in Fig. 2 by an arrow starting at the upper left corner of the matrix. As in Appendix I, each row of the matrix corresponds to a *block* on a working tape.

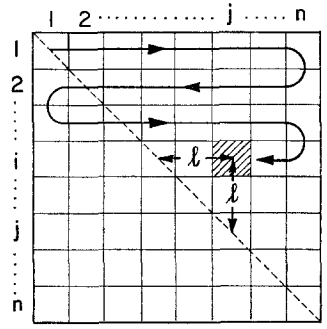

FIG. 2. Recognition matrix for a 2PDA1.

In the following, the heads H_1 and H_2, on T_1 and T_2, respectively, are usually moved in synchronism with each other in the same direction so that their relative positions with respect to the left ends of the tapes do not change. At times their relative positions do change, but then M can resort to the "reset" operation to recover the same relative positions. This can be done by moving both H_1 and H_2 towards the left end of T_1 and T_2, respectively. Whichever reaches the left end first stays within the first block and waits for the other head, scanning the first block back and forth. When the other head reaches the left end of its tape, the first head will be also on the left end. (To be more precise, their relative positions may defer by one square, but this apparent difficulty, is easily overcome by letting the first head go out of the first block by one square whenever necessary).

Before the execution of the algorithm, M marks the squares on T_1 and T_2, which correspond to the main diagonal of the recognition matrix. This can be easily done by scanning the input word n times, moving H_1 and H_2 to the right: When the input head scans the input word from left to right, the leftmost unchecked square is checked. At the same time H_1 and H_2 put marks on the squares they are currently scanning. When the input head scans the input tape from right to left, the leftmost unchecked square is checked and H_1 and H_2 put marks on their tapes. Here again the input head and H_1 and H_2 get "out of synchronization" by one square for each scan, but this can be easily overcome by a similar method to the one described in the previous paragraph. When all the squares corresponding to the main diagonal are marked, the heads H_1 and H_2 are "reset" again.

The following seven steps correspond to those of Aho et al. (pp. 201-202).

1. This can be done during the marking of the squares corresponding to the main diagonal.

2. This can be done without reversing H_1 or H_2. The input head is reversed n times. When this step is over H_1 and H_2 are "reset".

3. This step does not involve any reversal of H_1 or H_2. The input head is reversed up to n times, depending on the location of first element of the form $[0, (p, Z, q)]$.

4. Let the element found in step 3 be an element of $r(i, j)$. Now assume $d = 0$. Then the number of *blocks* H_2 has to cross to go to the jth block equals the number of *squares* H_1 has to cross before it can reach the square (in that block) which corresponds to $r(i, i)$. (This square has been marked before.) The number is indicated by l in Fig. 2. H_1 stays within the ith block scanning it from its left end to right end several times, each time checking a new square between and including the squares corresponding to $r(i, j)$ and $r(i, i)$. When

all these squares have been checked, H_2 will have reached the jth block. H_1 and H_2 can now be moved to the squares corresponding to $r(i, 1)$ and $r(1, j)$, respectively. The number of reversals each head undergoes is bounded by n.

The cases $d = \pm 1$ can be handled by moving H_1 to the next block (left or right, depending on the value of d), while letting H_2 stay in the jth block. When this step is over H_1 and H_2 are "reset".

5. Similar to step 3.
6. Easy to implement.
7. Go back to step 3.

Now we can count the total number of reversals needed to accept or reject an input word. Each of the steps 1-6 can be done with no more than $c_1 n$ reversals for some constant c_1 and step 7 doesn't involve any reversal. If the language involved is a 2DPDAl, then steps 3-7 are repeated at most $c_2 n$ times for some constant c_2 (p. 198). Thus our TM M is cn^2 reversal bounded for some constant c. If, on the other hand, the language is a 2PDAl, then the number of repetitions of steps 3-7 is bounded by $c_3 n^2$ for some constant c_3 (p. 202). Therefore M is cn^2 reversal bounded for some constant c.

RECEIVED: September 19, 1969; REVISED: March 30, 1970

ACKNOWLEDGMENT

The authors would like to thank the referee for his suggestions which have improved the presentation.

REFERENCES

1. A. V. AHO, J. E. HOPCROFT, AND J. D. ULLMAN, (1968), Tape and time complexity of pushdown automaton languages, *Information and Control* **13**, 186–206.
2. J. BEČVÁŘ, (1965), Real-time and complexity problems in automata theory, *Kybernetika* **6**, 475–498.
3. M. BLUM, (1967), A machine-independent theory of the complexity of recursive functions, *J. Assoc. Comput. Mach.* **14** (1967), 322–336.
4. P. C FISCHER, (1968), The reduction of tape reversals for off-line one-tape Turing machines, *J. Comput. Syst. Sci.* **2**, 136–147.
5. P. C. FISCHER, J. HARTMANIS AND M. BLUM, (1968), Tape reversal complexity hierarchies, *IEEE Conf. Record of the 9th Annual Symp. on Switching and Automata Theory*, 373–382.

6. S. GINSBURG, (1966), "The Mathematical Theory of Context-free Languages," McGraw-Hill, New York.
7. J. N. GRAY, M. A. HARRISON, AND O. IBARRA, (1967), Two-way pushdown automata, *Information and Control* **11**, 30–70.
8. J. HARTMANIS, (1968), Tape-reversal bounded Turing machine computations, *J. Comput. Syst. Sci.* **2**, 117–135.
9. J. HARTMANIS, (1968a), Computational complexity of one-tape Turing machine computations, *J. Assoc. Comput. Mach.* **15**, 325–339.
10. F. C. HENNIE, (1965), One tape off-line Turing machine computations, *Information and Control* **8**, 553–578.
11. F. C. HENNIE AND R. E. STEARNS, (1966), Two-tape simulation of multitape Turing machines, *J. Assoc. Comput. Mach.* **13**, 533–546.
12. T. KASAMI, (1967), A note on computing time for recognition of languages generated by linear grammars, *Information and Control* **10**, 209–214.
13. M. RABIN AND D. SCOTT, (1959), Finite automata and their decision problems, *IBM J. Res. Develop.* **3**, 115–125.
14. K. TANIGUCHI AND T. KASAMI, (1969), A note on computing time for the recognition of context-free languages by a single-tape Turing machine, *Information and Control* **14**, 278–284.
15. K. TORII, T. KASAMI, AND H. OZAKI, (1966), Recognition and syntax-analysis of context-free languages, *Report of Inst. Elec. Comm. Engrs. of Japan*, June.
16. D. YOUNGER, (1967), Recognition and parsing of context-free languages in time n^3, *Information and Control* **10**, 189–208.

Printed in Belgium